WHAT IS CLAIMED IS:

1. A processor comprising:

an execution unit to execute instructions;

a replay system coupled to the execution unit to replay instructions which have not executed properly, the replay system comprising:

a checker to determine whether each instruction has executed properly; and

a replay queue coupled to the checker to temporarily store one or more instructions ~~for replay.~~

2. The processor of claim 1 wherein said replay system further comprises:

a replay loop to route an instruction which executed improperly to an execution unit for replay; and

a replay queue loading controller to determine whether to load an improperly executed instruction to the replay loop or into the replay queue.

3. The processor of claim 2 and further comprising:

a scheduler to output instructions; and

a multiplexer or selection mechanism having a first input coupled to the scheduler, a second input coupled to the replay loop and a third input coupled to an output of the replay queue.

4. The processor of claim 1 wherein said replay queue comprises a replay queue coupled to the checker to temporarily store one or more long latency instructions until the instruction is ready for execution.

5. The processor of claim 1 wherein the replay queue comprises a replay queue coupled to the checker to temporarily store an instruction that is not ready to execute properly, the instruction being unloaded from the replay queue when the instruction is ready to execute properly.

6. The processor of claim 1 wherein the replay queue comprises a replay queue coupled to the checker to temporarily store an instruction in which source data must be retrieved from an external memory device, the instruction being unloaded from the replay queue when the source data for the instruction returns from the external memory device.

7. The processor of claim 1 wherein said execution unit is a memory load unit, the processor further comprising:

a first level cache system coupled to the memory load unit;

a second level cache system coupled to the first level cache system; and

wherein the memory load unit performs a data request to external memory if there is a miss on both the first level and second level cache systems.

8. The processor of claim 7 wherein a load instruction will be loaded into the replay queue when there is a miss on both the first level and second level cache systems, and the load instruction is unloaded from the replay queue for re-execution when the data for the instruction returns from the external memory.

9. A processor comprising:

a multiplexer having an output;

a scheduler coupled to a first input of the multiplexer;

an execution unit coupled to an output of the multiplexer;

a checker coupled to the output of the multiplexer to determine whether an instruction has executed properly;

a replay queue to temporarily store instructions, an output of the replay queue coupled to a second input of the multiplexer; and

a controller coupled to the checker to determine when to load an instruction into the replay queue and to determine when to unload the replay queue.

10. The processor of claim 9 and further comprising a staging section coupled between the checker and a third input to the multiplexer to provide a replay loop, the controller controlling the multiplexer to select either the output of the scheduler, the replay loop or the output of the replay queue.

11. The processor of claim 9 wherein the controller loads an instruction into the replay queue when the instruction is not ready to execute properly, and unloads the instruction from the replay queue when the instruction is ready to execute properly.

12. The processor of claim 9 wherein the controller determines when to unload the replay queue based on a data return signal.

13. A method of processing instructions comprising:

dispatching an instruction where the instruction to an execution unit and to a replay system;

determining whether the instruction executed properly;

if the instruction did not execute properly, then:

determining whether the instruction should be routed back for re-execution or whether the instruction should be temporarily stored in a queue.

14. A method of processing instructions comprising:

dispatching an instruction where the instruction is received by an execution unit and a replay system;

determining whether the instruction executed properly;

if the instruction did not execute properly, then:

routing the instruction to the execution unit for re-execution if the instruction is a first type of instruction;

otherwise, loading the instruction into a replay queue if the instruction is a second type of instruction.

15. The method of claim 14 wherein the first type of instruction comprises a short latency instruction, and the second type of instruction is a longer latency instruction.

16. A method of processing instructions comprising:

dispatching an instruction to an execution unit;

determining whether the instruction executed properly;

if the instruction did not execute properly, then:

determining whether an access across an external bus is required for proper instruction execution;

routing the instruction to the execution unit for re-execution if an external bus access is not required;

otherwise temporarily storing the instruction in a replay queue if an external bus access is required, then unloading the instruction from the replay queue to the execution unit for execution when the access across an external bus has been completed.

17. A method of processing instructions comprising:

dispatching a load instruction to a memory load unit;

determining whether the load instruction executed properly;

if the load instruction did not execute properly, then:

determining whether there was a cache miss to all on-chip cache systems;

routing the load instruction to the memory load unit for re-execution if there was not a miss to all on-chip cache systems;

otherwise, if there was a cache miss to all on-chip cache systems, then temporarily storing the load instruction in a replay queue and retrieving the data for the load instruction from external memory, and then unloading the load instruction from the replay queue to the memory load unit for re-execution when the data for the load instruction has been retrieved from the external memory.

18. A method of processing instructions comprising:

receiving an improperly executed instruction;

determining whether the instruction is an agent instruction;

if the instruction is an agent instruction, then loading the instruction into a replay queue;

otherwise, if the instruction is not an agent instruction, then loading the instruction into the replay queue if the following conditions are met:

a) there is already an agent instruction in the replay queue; and

b) the instruction is younger than the agent instruction in the replay queue; and

unloading one or more of the instructions in the replay queue to an execution unit when the agent instruction in the replay queue is ready to execute.

19. The method of claim 18 wherein said step of unloading comprises the step of unloading the instructions in the replay queue to an execution unit when data for the agent instruction is available to allow the instruction to execute properly.